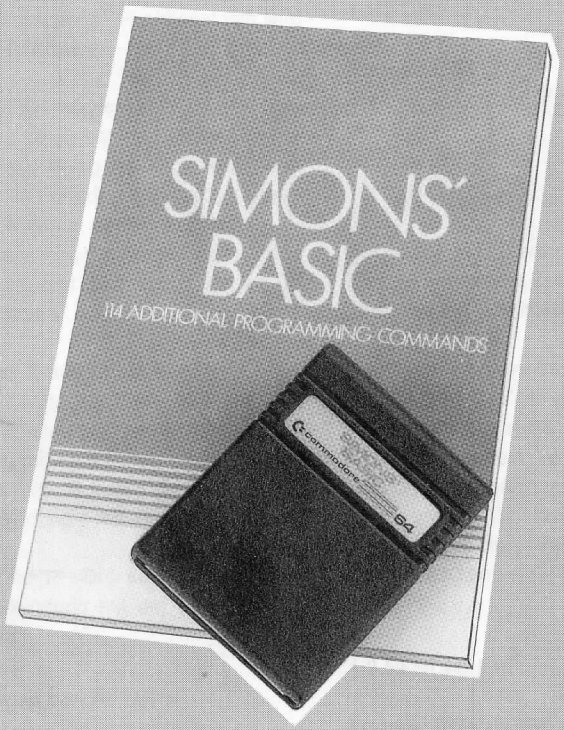
SIMONS´ BASIC

Best.-Nr. 95 11 88



CONRAD ELECTRONIC

SIMONS´ BASIC

Mit diesem Modul stehen dem Benutzer über 100 neue BASIC-Befehle zur Verfügung.

Jeder Befehl wird in dieser Anleitung erklärt.

1. Befehle für die Benutzung der Funktionstasten.

1.1 KEY

Format: KEY Nummer, "Befehl"
Ziel: Die entsprechende Funktionstaste wird mit einem Befehl belegt.

Der Befehl KEY ermöglicht es, die Funktionstasten mit den gewünschten Befehlen zu belegen und diese Belegung auch zu ändern. Die Nummer im Format des Befehls ist identisch mit der Nummer der Funktionstaste (1-16). Der Befehl muß in Anführungszeichen gesetzt werden und darf nicht mehr als 15 Zeichen umfassen.

Erreichen der Funktionstasten:
F1, F3, F5, F7 - Drücken der Funktionstasten
F2, F4, F6, F8 - Drücken der Funktionstasten in Verbindung mit Shift
F9, F11, F13, F15 - Drücken der Funktionstasten in Verbindung mit der Commodore-Taste
F10, F12, F14, F16 - Drücken der Funktionstasten in Verbindung mit der Commodore-Taste und Shift.

1.1.1 Anhängen von RETURN

Bei bestimmten Befehlen ist es sinnvoll, sie gleich mit RETURN abzuschließen, damit dieser Befehl sofort ausgeführt wird.

Dazu geht man wie folgt vor:
a) gewünschte Belegung einer Funktionstaste mit dem Befehl KEY eingeben, nicht mit RETURN abschließen.

b)+CHR$(13) anhängen und RETURN drücken. Der gewünschte Befehl wird nach dem Drücken der Funktionstaste angezeigt und gleich ausgeführt.

ACHTUNG:
c) Nach dem Ausschalten des Rechners sind die Funktionstastenbelegungen gelöscht!

1.1.2 DISPLAY

Format: DISPLAY
Ziel: Belegung der Funktionstasten anzeigen.

1.2 AUTO

Format: AUTO Startnummer, Schrittweite
Ziel: Automatische Zeilennumerierung mit vorgewählter Schrittweite.

Nach der Eingabe des Befehls AUTO erscheint die erste Zeilennummer auf dem Bildschirm. Nun kann die gewünschte BASIC-Zeile eingegeben werden; sie wird mit RETURN abgeschlossen. Es erscheint eine neue, um die Schrittweite erhöhte Zeilennummer. Um diese Funktion zu beenden, drückt man nach einer neuen Zeilennummer RETURN.

1.3 RENUMBER

Format: RENUMBER erste Zeilennummer , Schrittweite
Ziel: Umnumerierung der Zeilennummern bei Programmen.

RENUMBER übernimmt automatisch die Umnumerierung der Zeilennummern. Das Programm beginnt nach Ausführung des Befehls mit der ersten Zeilennummer, die eingegeben wurde. Die folgenden Zeilennummern sind jeweils um die eingegebene Schrittweite erhöht.
Dieser Befehl ist besonders sinnvoll, um in einem Programm Platz für weitere BASIC-Zeilen zu schaffen.

ACHTUNG:
RENUMBER verändert keine Sprungadressen der Befehle GOTO und GOSUB.

1.4 PAUSE

Format: PAUSE "Kommentar", Anzahl von Sekunden
oder: PAUSE Anzahl von Sekunden
Ziel: Verzögern der Programmausführung

PAUSE bietet die Möglichkeit, die Programmausführung an einer bestimmten Stelle für eine bestimmte Dauer zu verzögern. Diese Pause kann durch Drücken von RETURN abgebrochen werden. Zusätzlich besteht die Möglichkeit, einen Kommentar auf den Bildschirm zu bringen.

1.5 CGOTO

Format: CGOTO Ausdruck
oder: CGOTO Operand Operator Variable
Ziel: Sprung im Programm zu einer veränderbaren Zeilennummer.

Der CGOTO-Befehl ermöglicht es, in eine Zeile zu springen, die erst bei der Ausführung des Befehls berechnet wird.

1.6 RESET

Format: RESET Zeilennummer
Ziel: Setzen des DATA-Zeigers auf den ersten Wert einer bestimmten DATA-Zeile.

Im Standard-BASIC werden alle DATA-Anweisungen nacheinander gelesen. Es besteht nun die Möglichkeit, durch den Befehl RESTORE den DATA-Zeiger auf den ersten Wert der ersten DATA-Anweisung zu setzen.
RESET bietet die Möglichkeit, den DATA-Zeiger auf den ersten Wert einer bestimmten DATA-Zeile zu setzen. Beim nächsten READ-Befehl wird dieser Wert gelesen und der DATA-Zeiger um eine Stelle nach hinten versetzt.

1.7 MERGE

Format: MERGE "Programmname", Gerätenummer
Ziel: Laden eines vorher gespeicherten Programms in den Arbeitsspeicher des Rechners.

Ein im Arbeitsspeicher vorhandenes Programm bleibt erhalten. Dabei werden Programmname und Gerätenummer entsprechend dem LOAD-Befehl verwendet.
ACHTUNG:
Bei Verwendung des MERGE-Befehls dürfen in dem Programm, das geladen wird und dem Programm im Arbeitsspeicher keine gleichen Zeilennummern vorkommen. Es empfiehlt sich, das Programm im Arbeitsspeicher mit dem Befehl RENUMBER umzunumerieren.

1.8 Hilfen zum Listen der Programme

1.8.1 PAGE

Format: PAGE n
Ziel: Teilen eines Programms in Abschnitte zu n+1 Zeilen.

PAGE bietet die Möglichkeit, die Anzahl der Bildschirm-Zeilen vorzuwählen. Nachdem dieser Befehl ausgeführt wurde, zeigt der Befehl LIST genau n+1 Bildschirmzeilen des Programms an. Jeder weitere Abschnitt wird nach Drücken der RETURN-Taste gelistet.
Bei n=0 hat der PAGE-Befehl keine Wirkung auf den LIST-Befehl. Das Listen kann mit der RUN/STOP-Taste unterbrochen werden.
ACHTUNG:
Der Parameter "n" bezieht sich auf Bildschirmzeilen, nicht auf Programmzeilen.

1.8.2 OPTION

Format: OPTION n
Ziel: Hervorheben aller neuen Befehle des SIMONS´ BASIC.

OPTION mit dem Parameter n=10 bewirkt eine reverse Darstellung der SIMONS´ BASIC-Befehle, wenn das Programm gelistet wird. Ein beliebiger Parameterwert schaltet das Hervorheben wieder ab.

1.8.3 DELAY

Format: DELAY n
Ziel: Verändern der Geschwindigkeit, mit der ein Programm gelistet wird.

Während der Ausführung von LIST besteht die Möglichkeit, die Geschwindigkeit, mit der die Zeichen auf den Bildschirm kommen, zu verändern. Hierzu gibt man vor dem LIST-Befehl den Befehl DELAY n (RETURN) ein. Der Parameter n kann Werte von 1 bis 255 annehmen; n=1 höchste Geschwindigkeit, n=255 niedrigste Geschwindigkeit.
Nach LIST (RETURN) wird das Programm mit der gewünschten Geschwindigkeit gelistet, wenn die SHIFT-Taste gedrückt wird. Weiterhin besteht die Möglichkeit, den LIST-Vorgang durch Drücken der CTRL-Taste zu verlangsamen oder durch Betätigen der Commodore-Taste zu unterbrechen.

2.0 FIND

Format: FIND BASIC-Code
oder: FIND Character String
Ziel: Suchen eines BASIC-Codes oder Character Strings und Anzeigen der Zeilennummern, in denen der Code bzw. String steht. Der Befehl FIND durchsucht das Programm nach dem angegebenen BASIC-Code oder dem Character String und zeigt auf dem Bildschirm die Zeilennummern an, in denen das Gesuchte steht.
ACHTUNG:
Jede Leerstelle (SPACE) nach FIND wird als Zeichen angesehen, nach dem gesucht werden soll.

2.1 Hilfen zur Fehlerbeseitigung

2.1.1 TRACE

Format: TRACE 10
oder:TRACE 0
Ziel: Anzeige der Zeilennummer, die gerade abgearbeitet wird.

Der Befehl TRACE 10 wird eingegeben, bevor das Programm gestartet wird. Wenn das Programm ausgeführt wird, erscheint in der rechten oberen Ecke ein Fenster, in dem die sechs letzten Zeilennummern angezeigt werden. Die Zeilennummern rollen automatisch, so daß die unterste immer der zuletzt ausgeführten BASIC-Zeile entspricht.
Die Zeilennummern werden im HiRes- oder Multicolor-Modus nicht sichtbar.
Das Fenster überdeckt den normalen Bildschirminhalt.
Der TRACE-Befehl wird durch die Eingabe von TRACE 0 beendet.

2.1.2 RETRACE

Format: RETRACE
Ziel: Nochmalige Anzeige des TRACE-Fensters.

Wird im TRACE-Modus ein Programm unterbrochen und der Bildschirm gelöscht, dann kann das Fenster mit den letzten Zeilennummern durch den Befehl RETRACE nochmals dargestellt werden.

2.2 DUMP

Format: DUMP
Ziel: Listet die Variablen eines Programms und Ihre momentanen Werte.

Nach dem Befehl DUMP werden alle Variablen mit ihren momentanen Werten in der Reihenfolge, in der sie im Programm vorkommen, gelistet. Es besteht die Möglichkeit, das Listen durch Drücken der CTRL-Taste zu verzögern.

2.3 COLD

Format: COLD
Ziel: Bringt den Commodore in den Ausgangszustand.

Nach COLD (RETURN) wird der Computer in den Ausgangszustand von SIMONS´ BASIC gesetzt und meldet sich mit dem normalen Einschaltbild.
ACHTUNG:
Das Programm im Arbeitsspeicher geht verloren. Es kann nur gerettet werde, wenn man, bevor eine neue BASIC-Zeile eingegeben wird, den Befehl OLD (siehe Abschnitt 2.5.) ausführt.

2.4 Hilfen zum Programmschutz

2.4.1 DISAPA

Format: DISAPA:
Ziel: Anzeigen, daß die Anweisung in der Programmzeile geschützt werden soll.

Der DISAPA-Befehl wird als Befehl in einer Programmzeile verwendet, um anzuzeigen, daß diese geschützt werden soll. Er setzt automatisch vier Doppelpunkte (:) vor den nachfolgenden Befehl.

2.4.2 SECURE

Format: SECURE 0
Ziel: Schützen aller Programmzeilen, in denen der DISAPA-Befehl vorkommt.

Der SECURE-Befehl bewirkt, daß der Code in einer Programmzeile, der hinter dem DISAPA-Befehl steht, geschützt wird. Die Programmzeile wird normal ausgeführt.

2.5 OLD

Format: OLD
Ziel: Aufheben des Befehls NEW.

OLD ermöglicht es, ein Programm, das mit dem Befehl NEW gelöscht wurde, zurückzuholen und auszuführen (es werden die Zeiger START OF BASIC und END OF BASIC auf den ursprünglichen gesetzt).
ACHTUNG:
Der Befehl ist nur wirksam, solange noch keine neue BASIC-Zeile eingegeben wurde.

2.6 Zeichenketten

2.6.1 INSERT

Format: INSERT ("einzusetzender String", "äußerer String", p)

Ziel: Einsetzen eines Strings in einen anderen.

INSERT ermöglicht es, einen String in einen anderen einzubauen. Dabei entsteht ein neuer String.
Der Parameter p gibt die Position an, ab welcher der String eingebaut werden soll.
Der einzusetzende String und der äußere String können sowohl als Stringvariable als auch in Anführungszeichen angegeben werden.
Der INSERT-Befehl kann zusammen mit logischen Vergleichsoperatoren benutzt werden.
ACHTUNG:
Der Parameter p muß kleiner sein als die Länge des äußeren Strings. Andernfalls folgt die Fehlermeldung "?INSERT TOO LARGE IN Zeile x".
Der neu entstandene String darf nicht länger als 255 Zeichen sein, da sonst die Fehlermeldung "?STRING TOO LARGE IN Zeile x" folgt.

2.6.2 INST

Format: INST ("überschreibender String", "alter String", p)
Ziel: Überschreibt einen String ab einer bestimmten Position.

INST überschreibt einen Teil des äußeren Strings mit einem String. Der Parameter p gibt die Position an, ab welcher der äußere String überschrieben wird. Dabei kann der neu entstandene String länger als der alte, darf jedoch höchstens 255 Zeichen sein.

2.6.3 PLACE

Format: PRINT PLACE ("gesuchter String", "Zeichenkette")
Ziel: Suchen eines bestimmten Strings.

PLACE sucht in einer Zeichenkette nach einem bestimmten String. Als Ergebnis wird eine INTEGER-Zahl geliefert, welche die Position des ersten gesuchten Zeichens beschreibt. Wird der gesuchte String nicht gefunden, so ist das Ergebnis 0.

2.6.4 DUP

Format: DUP ("Zeichenkette", n)
oder: DUP (Stringvariable, n)
Ziel: Vervielfachen einer Zeichenkette

Der DUP-Befehl vervielfacht eine Zeichenkette n-mal. Dabei entsteht ein neuer String, der aus n Zeichenketten besteht.
ACHTUNG:
n muß so gewählt werden, daß der neue String nicht länger als 255 Zeichen ist (sonst Programmabsturz).

2.6.5 CENTRE

Format: CENTRE "Zeichenkette"
oder: CENTRE Stringvariable
Ziel: Zentrieren einer Zeichenkette in einer Bildschirmzeile.

CENTRE bringt eine Zeichenkette genau in die Mitte einer Bildschirmzeile.
Die Zeichenkette darf dabei nicht länger als 39 Zeichen sein.

2.6.6 USE

Format: USE "###.###", "Zeichenkette" : PRINT
oder: USE "## text . ### text", "Zeichenkette": PRINT
oder: USE Stringvariable, Stringvariable: PRINT
Ziel: Formatierte Ausgabe von numerischen Werten.

USE ermöglicht es, einen numerischen Wert in einem vorgegebenen Format auszugeben.
Jedes "Nummer"-Zeichen (#) gibt dabei eine Stelle vor bzw. nach dem Komma (Punkt) an. Es ist auch möglich, Text in eine formatierte Zeile einzubauen. Diese Formatangabe kann auch in Form einer Stringvariablen dargestellt werden.
Die Zahl, welche ausgegeben werden soll, muß als Stringvariable oder Zeichenkette dargestellt werden, d.h., Zahlen müssen vor der USE-Ausgabe mit dem Befehl STR$ in einen String umgewandelt werden.

2.6.7 AT

Format: PRINT AT (s,z) "Zeichenkette"
oder: PRINT "1.String" AT (s,z) "2.String"
Ziel: Positionierte Ausgabe eines Strings.

AT positioniert die Ausgabe eines Strings auf eine bestimmte Bildschirmposition.
Es gibt dabei die Spalte (0-39) und z die Zeile (0-24) an. Es ist möglich, den AT-Befehl mehrmals hintereinander zu gebrauchen.

2.7 Kontrolle der Eingabe

2.7.1 FETCH

Format: FETCH "Kontrolltaste", 1 , Variable.
Ziel: FETCH ermöglicht es, bei der Eingabe von der Tastatur nur bestimmte Zeichen zuzulassen und auch die Anzahl der Zeichen zu begrenzen.

Die nach dem Anführungszeichen eingegebene Kontrolltaste bestimmt dabei, welches Zeichen eingegeben werden kann; 1 gibt die Anzahl an, wieviele Zeichen höchstens eingegeben werden sollen. Der eingegebene String oder numerische Wert wird unter einer Variablen abgelegt, deren Namen angegeben werden muß.

Kontrolltaste: zugelassene Zeichen
CLR/HOME: nur Großbuchstaben (CHR$(65) bis CHR$(90))
CURSOR DOWN: CHR$(32) bis CHR$(64) (Zahlen, Satzzeichen)
CURSOR RIGHT: nur Großbuchstaben und ihre Funktion mit SHIFT.

Zusätzlich ist es möglich, die Eingabe auf ganz bestimmte Zeichen zu begrenzen, indem man statt der Kontrolltaste genau die gewünschten Zeichen eingibt.

2.7.2 INKEY

Format: INKEY
Ziel: Abfrage, welche Funktionstaste gedrückt wurde.

INKEY ermöglicht es, festzustellen, welche Funktionstaste gedrückt wurde. Das Ergebnis ist die Nummer der Funktionstaste (1-16) oder 0 falls keine Funktionstaste gedrückt wurde. Die INKEY-Abfrage muß in ein Programm eingebunden werden.

2.8 Zusätzliche arithmetische Operatoren

2.8.1 MOD

Format: MOD (x,y)
Ziel: Bestimmt den Rest nach einer Integer-Division.
Der MOD-Befehl wandelt die Parameter x und y zunächst in zwei Integerzahlen um (ohne Aufrundung). Danach wird eine Integerdivision durchgeführt. Das Ergebnis ist eine Integerzahl, die den Rest der Division darstellt. Der Befehl kann direkt oder im Programm durchgeführt werden.

2.8.2 DIV

Format: DIV (x,y)
Ziel: Bestimmung der Vorkommazahl bei einer Integer-Division.

Der Befehl DIV wandelt die Parameter x und y in zwei Integerzahlen und führt eine Division aus. Das Ergebnis ist eine Integerzahl, die Vorkommazahl. Der Befehl kann direkt oder im Programm verwendet werden.

2.8.3 FRAC

Format: FRAC (n)
Ziel: Bestimmung der Nachkommazahl.

Der Befehl FRAC bestimmt die Nachkommazahlen einer Gleitkommazahl.
Dabei werden maximal 9 Stellen hinter dem Komma angegeben.

2.9 Zahlenumwandlung

2.9.1 % - Binär in Dezimal

Format: PRINT % Binärzahl
Ziel: Umrechnung einer Binärzahl in eine Dezimalzahl.

Der %-Befehl wandelt die angegebene Binärzahl in eine Dezimalzahl. Die Binärzahl muß 8 Ziffern lang sein und darf nur die Ziffern 0 und 1 beinhalten. Ist dies nicht der Fall, so folgt die Fehlermeldung:
"?NOT BINARY CHAR".

2.9.2 $ - Hexadezimal in Dezimal

Format: PRINT $ Hexadezimalzahl
Ziel: Umwandlung einer Hexadezimalzahl in eine Dezimalzahl.

Der $-Befehl wandelt die angegebene Hexadezimalzahl in eine Dezimalzahl.
Die Hexadezimalzahl umfaßt 4 Ziffern, die aus den Werten 0-F (0-15) bestehen können. Werden diese Bedingungen nicht eingehalten, so folgt die Fehlermeldung:
"?NOT HEX CHARACTER".

3.0 EXOR

Format: EXOR (n, n1)
Ziel: Eine Exklusiv-oder-Operation ausführen.

Der Befehl EXOR ermöglicht es, eine Exklusiv-oder-Operation zwischen zwei Zahlen durchzuführen. Die angegebenen Zahlen werden in Binärzahlen umgewandelt und dann Bit für Bit verknüpft. Das Ergebnis wird in eine Dezimalzahl umgewandelt. Die beiden Parameter n und n1 können Werte zwischen 0 und 65535 annehmen. Es können auch Zahlen verschiedener Zahlensysteme verknüpft werden.

3.1 Diskettenbefehle

3.2 DISK

Format: DISK, "Anweisung"
Ziel: Öffnen des Diskettenkanals, Anweisung ausführen, Diskettenkanal schließen.

Der Befehl DISK ersetzt folgende BASIC-Befehle:
OPEN, logische Filenummer, Gerätenummer, Sekundäradresse:
PRINT # logische Filenummer. Er öffnet einen Kanal zu einer Disketteneinheit, führt den angegebenen Befehl aus und schließt den Kanal wieder.

3.3 DIR

Format: DIR "$
oder: DIR "$:String*
oder: DIR "$:?String*
Ziel: Listen der Directory oder eines Teils der Directory.

Der DIR-Befehl ersetzt den Befehl LOAD"$",8. Dabei hat man die Möglichkeit, die gesamte Directory zu listen, oder nur bestimmte Filenamen. Jedes Fragezeichen steht für einen beliebigen Buchstaben. Der Stern kürzt den String ab.

3.4 Grafik mit SIMONS´ BASIC

3.5 Einführung

3.5.1 Bildschirmaufbau

Bei der Verwendung der Bildschirmgrafik wird der Bildschirm des Commodore in eine Matrix von maximal 320 x 200 Punkten aufgeteilt. Jeder Punkt hat eine x- und y-Koordinate. Die linke obere Ecke hat die Koordinaten 0,0 (x,y). Die High-Resolution-Grafik teilt den Bildschirm in eine Matrix von 320 (x) und 200 (y) Punkten. Die Multi-Color-Grafik teilt ihn in 160 x 200 Punkte; der Abstand zweier Punkte in x-Richtung ist also doppelt so groß wie bei der High-Resolution-Grafik.

3.5.2 Commodore Farbe

Der Commodore bietet eine Auswahl von 16 verschiedenen Farben. In einer Matrix von 8 x 8 Punkten können bis zu drei verschiedene Farben gewählt werden. Jeder Farbe ist eine Ziffer zugeordnet:

0 = schwarz	8 = orange
1 = weiß	9 = braun
2 = rot	10 = hellrot
3 = türkis	11 = grau 1
4 = violett	12 = grau 2
5 = grün	13 = hellgrün
6 = blau	14 = hellblau
7 = gelb	15 = grau 3

3.5.3 Zeichentyp

Alle Zeichen-Befehle des SIMON's BASIC besitzen ein gemeinsames Merkmal: den Zeichentyp. Dieser Zeichentyp bestimmt, wie ein Punkt auf dem Bildschirm dargestellt wird.

Die Bedeutung des Zeichentyps:

a) HIGH RESOLUTION MODE

Zeichentyp	Funktion
0 –	löscht einen Punkt
1 –	zeichnet einen Punkt
2 –	invertiert einen Punkt (ist ein Punkt da, so wird er gelöscht und umgekehrt)

b) MULTI-COLOR-MODE

Zeichentyp	Funktion
0 –	löscht einen Punkt
1 –	zeichnet einen Punkt in COL 1
2 –	zeichnet einen Punkt in COL 2
3 –	zeichnet einen Punkt in COL 3
4 –	invertiert einen Punkt. Hintergrundfarbe wird in COL 3, COL 1 in COL 2, COL 2 in COL 1 und COL 3 in Hintergrundfarbe dargestellt.

3.6 Grafik-Befehle

3.6.1 HiRes

Format: HiRes zf, hg
Ziel: Bestimmen der Zeichen- und Hintergrundfarbe

Der **HiRes**-Befehl schaltet den Bildschirm auf hochauflösende Grafik. Jeder Punkt des Bildschirmes kann nun einzeln gesetzt werden (320 x 200 Bildschirm-Matrix). Der Parameter zf bestimmt die Farbe, in der ein Punkt gesetzt wird. Der Parameter hg bestimmt die Farbe des Hintergrundes, in der ein Punkt gesetzt wird.

3.6.2 REC

Format: REC x , y , x1 , y1 , Zeichentyp
Ziel: Ein Rechteck zeichnen.

Der REC-Befehl zeichnet ein Rechteck auf den Bildschirm. Die beiden ersten Parameter x und y bestimmen die Koordinaten der linken oberen Ecke des Rechtecks. Die Parameter x1 und y1 bestimmen die Seitenlängen des Rechtecks.
Der Zeichentyp ist in Abschnitt 3.5.3 beschrieben.

3.6.3 MULTI

Format: HiRes zf, hg : MULTI c1, c2, c3
Ziel: Den Multi-Color-Modus einschalten und drei Zeichenfarben bestimmen.

Kommt nach dem **HiRes**-Befehl der Befehl MULTI, so wird die Multi-Color-Grafik eingeschaltet. Ein gezeichneter Punkt hat hier die doppelte Breite gegenüber einem Punkt in **HiRes**-Grafik (160 x 200 Bildschirm-Grafik). Die Parameter c1 bis c3 legen die drei Farbmöglichkeiten (siehe Farbtabelle in Abschnitt 3.5.2) fest, die dann durch den Zeichentyp die Farbe eines Punktes bestimmen.

3.6.4 LOW COL

Format: LOW COL c1 c2, c3
Ziel: Farbwechsel

Der Befehl LOW COL ermöglicht es, drei weitere Farben für einen Zeichenbefehl bereitzustellen.
Die Parameter c1 bis c3 bestimmen diese Farben, entsprechend dem Befehl MULTI.
ACHTUNG:
Es müssen immer alle drei Parameter angegeben werden, auch in **HiRes**-Grafik.

3.6.5 HI COL

Format: HI COL
Ziel: Zurückholen der unter MULTI festgelegten Farben.

Nach dem HI COL-Befehl können die unter MULTI festgelegten Farben wieder verwendet werden (in Verbindung mit dem Zeichentyp).

3.6.6 PLOT

Format: PLOT x, y, Zeichentyp
Ziel: Einen Punkt zeichnen.

Der Befehl PLOT zeichnet einen Punkt auf den Bildschirm. Die Parameter x und y geben die Koordianten des Punktes an. Der Zeichentyp ist in Abschnitt 3.5.3 beschrieben.

3.6.7 TEST

Format: Variable = TEST (x , y)
Ziel: Die Lage eines Punktes bestimmen.

Der Befehl TEST bestimmt die Lage eines Punktes auf dem Bildschirm.
Die Parameter x und y bestimmen die Bildschirmkkoordinaten des TEST-Punktes. Das Ergebnis in der Variablen ist 1, wenn an dieser Stelle ein Punkt gesetzt ist, im anderen Fall ist das Ergebnis 0.

3.6.8 LINE

Format: LINE x, y, x1, y1, Zeichentyp
Ziel: Eine Linie zeichnen.

Der Befehl LINE zeichnet eine Linie. x und y sind die Anfangskoordinaten, x1 und y1 die Endkoordinaten. Der Zeichentyp ist in Abschnit 3.5.3 beschrieben.

3.6.9 CIRCLE

Format: CIRCLE x, y, x1, y1, Zeichentyp
Ziel: Zeichnen einer Ellipse.

Mit dem Befehl CIRCLE ist es möglich, eine Ellipse zu zeichnen. Die Parameter x und y bestimmen die Mittelpunkt-Koordinaten, x1 den Radius in x-Richtung, y1 den Radius in y-Richtung. Der Zeichentyp ist in Abschnitt 3.5.3 beschrieben.

3.6.10 ARC

Format: ARC x, y, sa, ea, i, x1, y1, Zeichentyp
Ziel: Einen Bogen zeichnen.

Der ARC-Befehl ermöglicht es, einen Bogen zu zeichnen. Die Parameter haben folgende Funktion:
x , y = Mittelpunkt-Koordinaten der Ellipse (Kreis), von der ein Bogen gezeichnet werden soll.
sa = Startwinkel
se = Endwinkel
i = Abstand der Winkel, die berechnet werden
Startwinkel + i = nächster Punkt, der berechnet wird. Diese beiden Punkte werden linear verbunden.
x1 = x-Radius der Figur, von welcher der Bogen gezeichnet werden soll.
y1 = y-Radius der Figur, von welcher der Bogen gezeichnet werden soll.

Der Zeichentyp ist in Abschnitt 3.5.3 beschrieben.

3.6.11 ANGL

Format: ANGL x, y, Winke , x1, y1, Zeichentyp
Ziel: Einen Radius einer Figur zeichnen.

Der ANGL-Befehl ermöglicht es, einen Radius in eine Ellipse einzuzeichnen. Die Ellipse muß dafür nicht auf dem Bildschirm vorhanden sein. Die Parameter haben folgende Funktion:
x , y = Mittelpunkt der gedachten Ellipse ist Startpunkt des Radius'.
Winkel = Winkel, der unter dem Radius steht.
x1 , y1 = Halbachsen der Ellipse, von welcher der Radius gezeichnet werden soll.

3.6.12 PAINT

Format: PAINT x, y, Zeichentyp
(nur 0, 1, 2 o. 3)
Ziel: Eine geschlossene Fläche mit einer Farbe ausfüllen.

Der PAINT-Befehl füllt eine umschlossene Fläche mit einer Farbe aus. Die Farbe wird durch den Zeichentyp bestimmt. Die Parameter x und y geben die Koordinaten eines Punktes innerhalb der zu färbenden Fläche an. Eine Fläche kann mehrmals mit einer Farbe gefüllt werden.

3.6.13 BLOCK

Format: BLOCK x, y, x1, y1, Zeichentyp
Ziel: Ein ausgefülltes Rechteck.

Zusätzlich zu der Möglichkeit, ein Rechteck zu zeichnen und dies mit einer Farbe zu füllen, kann mit dem Befehl BLOCK ein ausgefülltes Rechteck gezeichnet werden.

Entscheidend ist dabei die Zeichengeschwindigkeit.
Die Parameter x und y bestimmen die Koordinaten der linken oberen Ecke, x1 begrenzt den Block in x-Richtung, y1 begrenzt ihn in y-Richtung.
Der Zeichentyp ist in Abschnitt 3.5.3 beschrieben.

3.6.14 DRAW

Format: DRAW "nnnn...9", x, y, Zeichentyp
oder: DRAW Stringvariable, x , y , Zeichentyp
Ziel: Eine Figur zeichnen.

Der DRAW-Befehl ermöglicht es, eine beliebige Figur zu entwerfen und diese auf dem Bildschirm zu zeichnen. Die Figur entsteht ähnlich wie das Zeichnen auf dem Papier. Man gibt durch Ziffern (n) an, in welche Richtung sich der "Zeichenstift" bewegt und ob er sichtbar zeichnet oder nicht. Bedeutung der Ziffern:

0 = Bewegung nach rechts, nicht sichtbar

1 = Bewegung nach oben, nicht sichtbar

2 = Bewegung nach unten, nicht sichtbar

3 = Bewegung nach links, nicht sichtbar

4 = 2

5 = Bewegung nach rechts, sichtbar

6 = Bewegung nach oben, sichtbar

7 = Bewegung nach unten, sichtbar

8 = Bewegung nach links, sichtbar

Die Ziffern werden in Anführungszeichen direkt angegeben oder vorher als String festgelegt. Die Parameter x und y geben die Startkoordinaten an.
Der Zeichentyp ist in Abschnitt 3.5.3 beschrieben.

3.6.15 ROT

Format: ROT r, s
Ziel: Eine Figur verdrehen und die Größe verändern.

Der Befehl ROT ermöglicht es, eine Figur, die mit dem DRAW-Befehl entworfen wurde, um einen bestimmten Winkel zu drehen. Dieser Winkel wird durch den Parameter "r" bestimmt. Er muß im Bereich von 0 bis 7 liegen. Der zweite Parameter bestimmt die Größe der Figur; er kann zwischen 0 und 255 liegen.
ACHTUNG:
Die Größe der Figur hängt von der gewählten Grafik ab.
HiRes = 320 x 200 Punkte
MULTI =160 x 200 Punkte

3.6.16 CSET

Format: CSET n
Ziel: Auswahl eines Zeichensatzes oder Zurückholen der letzten Grafik.

Der CSET-Befehl ermöglicht es, den Zeichensatz des Commodore im Programm umzuschalten.
CSET 0 = Großschrift / Grafik-Modus
CSET 1 = Großschrift- / Kleinschrift-Modus
Weiterhin kann mit diesem Befehl die zuletzt abgebildete Grafik nochmals dargestellt werden.
CSET 2 = **HiRes**-Grafik
CSET 2 : MULTI c1,c2,c3 = MULTI-Color-Grafik
Die Parameter c1 bis c3 bestimmen die neuen Farben.

3.7 Text in einer Grafik

3.7.1 CHAR

Format: CHAR x, y, Bildschirmcode, Zeichentyp, Größe
Ziel: Ein einzelnes Zeichen auf einen Grafik-Bildschirm bringen.

Der CHAR-Befehl ermöglicht es, Zeichen auf einem Grafik-Bildschirm darzustellen. Die Parameter haben folgende Funktion:
x, y = Koordinaten, welche die linke obere Ecke des Zeichens positionieren.
Bildschirmcode = Zeichen, das gePOKEd wird.
Zeichentyp = siehe Abschnitt 3.5.3.
Größe = Faktor, um den sich das Zeichen in Vertikalrichtung ausdehnt.

3.7.2 TEXT

Format: TEXT x, y, "(CTRL-A) String", Zeichentyp, Größe, Abstand
oder: TEXT x, y, "(CTRL-B) String", Zeichentyp, Größe, Abstand
Ziel: Eine Zeichenkette auf einen Grafik-Bildschirm bringen.

Der Befehl TEXT ermöglicht es, Zeichenketten auf einem Grafik-Bildschirm darzustellen. Die Parameter haben folgende Funktion:
x , y = Koordinaten, welche die linke obere Ecke des Strings positionieren. Der nächste Parameter ist der String selbst oder eine Variable. Das erste Zeichen bestimmt, ob der String im Groß- oder Kleinschrift-Modus dargestellt wird.
CTRL-A = Großschrift = Invers A
CTRL-B = Kleinschrift = Invers B
Zeichentyp = siehe Abschnitt 3.5.3.
Größe = Faktor, um den sich der String in Vertikalrichtung ausdehnt.
Abstand = bestimmt den Abstand zwischen den einzelnen Zeichen.

3.8 Bildschirmsteuerung

3.9 FLASH

Format: FLASH f1, speed
Ziel: Blinken einer Bildschirmfarbe.

Der Befehl FLASH bewirkt einen dauernden Wechsel einer bestimmten Farbe zwischen Invers- und Normalschrift.
Der Parameter f1 bestimmt die Farbe, speed die Geschwindigkeit. Speed kann Werte zwischen 0 und 255 annehmen.
Die Einheit ist 1/16 s.
Der Befehl FLASH ist bei HiRes- oder Multi-Color-Grafik unwirksam.

4.0 OFF

Format: OFF
OFF beendet den FLASH-Befehl.

4.1 BFLASH

Format: BFLASH speed, f1, f2
Ziel: Farbwechsel des Bildschirmrahmens.

Der Befehl BFLASH bewirkt einen dauernden Farbwechsel des Bildschirmrahmens. Die Parameter f1 und f2 bestimmen die Farben. Der Parameter speed bestimmt die Geschwindigkeit des Farbwechsels (1-255).

4.2 BFLASH 0

Format: BFLASH 0
Der Befehl BFLASH 0 beendet den Befehl BFLASH.

4.3 FCHR

Format: FCHR r, c, w, d, Code

Ziel: Einen Bereich des Bildschirms mit einem Zeichen füllen.

Der Befehl FCHR ermöglicht es, einen bestimmten Bereich des Bildschirms mit einem bestimmten Zeichen zu füllen. Die Parameter r (0-24) und c (0-39) bestimmen die linke obere Ecke des Feldes, w die Anzahl der Spalten und d die Anzahl der Zeilen, Code ist der "POKE-Code" des gewünschten Zeichens.

4.4 FCOL

Format: FCOL r, c, w, d, Farbe
Ziel: Die Zeichenfarbe in einem bestimmten Bereich bestimmen.

Der Befehl FCOL legt die Zeichenfarbe eines bestimmten Bildschirmbereiches fest. Die Parameter r, c, w, d entsprechen den Paramtern des Befehls FCHR (Abschnitt 4.3), der letzte Parameter bestimmt die Farbe.

4.5 FILL

Format: FILL r, c, w, d, Code, Farbe
Ziel: Einen Bereich des Bildschirms mit einem bestimmten Zeichen in einer bestimmten Farbe füllen.

Der Befehl FILL füllt einen Bereich des Bildschirms mit einem Zeichen in vorgegebener Farbe. Die Parameter r, c, w, d, Code entsprechen den Parametern des Formats FCHR (Abschnitt 4.3), Farbe bestimmt die Zeichenfarbe.

4.6 MOVE

Format: MOVE r, c, w, d, dr, dc
Ziel: Duplizieren eines Bildschirmbereiches.

Der MOVE-Befehl dupliziert einen Bildschirmbereich.
Die Parameter r, c, w, d geben den Bereich an, der dupliziert werden soll, entsprechend Abschnitt 4.3 Die Parameter dr und dc geben die absolute Position an, ab welcher der Bereich dupliziert wird.

4.7 INV

Format: INV r, c, w, d
Ziel: Einen Bildschirmbereich invertieren.

Der Befehl INV invertiert alle Zeichen in einem bestimmten Bildschirmbereich.
Die Parameter r, c, w, d entsprechen den Parametern in Abschnitt 4.3.

4.8 Bildschirmrollen

Format: Richtung W, sr, sc, er, ec
oder: Richtung B, sr, sc, er, ec
Ziel: Rollen eines bestimmten Bildschirmbereiches.

SIMON's BASIC bietet die Möglichkeit, bestimmte Bildschirmbereiche in vier verschiedene Richtungen zu rollen.

Der erste Parameter bestimmt die Richtung des Bildschirmrollens :
LEFT, RIGHT, UP, DOWN.

Der zweite Parameter kann entweder W oder B sein.
W = Zyklisches Bildschirmrollen, d.h., es geht keine Zeile verloren.
B = Bildschirmrollen ohne Bildumlauf, d.h. es werden Leerzeilen bzw. Spalten nachgeschoben. Die Parameter sr, sc, er und ec bestimmen den Bildschirmausschnitt, der gerollt wird.
sr/sc = erste Zeile/Spalte
er/ec = letzte Spalte/Zeile

4.9 Abspeichern und Laden von Bildschirmdaten

4.9.1 SCRSV

Format: SCRSV 2,8,2,"Name,s,w"
oder: SCRSC 1,1,1,"Name,s,w"
Ziel: Abspeichern eines Low-Resolution-Bildschirmes.

Der Befehl SCRSV ermöglicht es, Bildschirmdaten direkt auf Diskette oder Kassette abzuspeichern. Die zweite Ziffer hinter dem Befehl gibt die entsprechende Gerätenummer an. Der Name bestimmt den Filenamen, unter dem der Bildschirminhalt abgespeichert wird und unter dem die Daten mit dem Befehl SCRLD (siehe Abschnitt 4.9.2) wieder geladen werden können. Mit diesem Befehl können keine Hi-Res- oder Multi-Color-Grafiken abgespeichert werden.

4.9.2 SCRLD

Format: SCRLD 2,8,2,"Name"
oder: SCRLD 1,1,1,"Name"
Ziel: Laden eines Low-Resolution-Bildschirmes.

Mit diesem Befehl können Bildschirmdaten, die mit dem Befehl SCRSV gespeichert wurden, wieder direkt geladen werden. Die Parameter sind entsprechend dem Befehl SCRSV anzugeben.

5.0 Ausdruck von Bildschirmdaten

5.0.1 COPY

Format: COPY
Ziel: Der Befehl COPY führt eine Hardcopy einer **HiRes**- oder Multi-Color-Grafik aus.
Um auf einem Drucker genau einen Kreis zu bekommen, müssen der x- und y-Radius des Befehls CIRCLE gleich sein.

5.0.2 HRDCPY

Format: HRDCPY
Ziel: Dieser Befehl führt eine Hardcopy eines Low-Resolution-Bildschirmes aus.

Man hat die Möglichkeit, Daten auf dem Bildschirm auszugeben und dann mit einem Befehl auf dem Drucker.

5.1 Sprite und Grafik

5.2 Gestalten eines MOBs

5.2.1 DESIGN

Format: DESIGN c, ad
oder: DESIGN c, sa + gc
Ziel: Zuteilung von Speicherplatz für einen MOB.

Der DESIGN-Befehl reserviert den benötigten Speicherplatz im Speicher des Rechners für einen MOB, der konstruiert werden soll. Der erste Parameter gibt an, ob der MOB ein high-resolution (0) oder ein multi-color-MOB (1) ist.
Der zweite Parameter definiert die Startadresse im Speicher des Computers. Diese Startadresse errechnet sich aus einer Block-Nummer multipliziert mit 64, da jeder MOB 64 Byte Speicherplatz benötigt. Wenn ein MOB auf einem High-Resolution- Bildschirm dargestellt werden soll, so muß zu der Startadresse ein konstanter Wert von 49152 ($C000) addiert werden.
Wird im Programm der Befehl MEM (siehe Abschnitt 5.2.1) verwendet, so können nur Block 192 bis 255 für MOBs benutzt werden.
ACHTUNG:
Es können so viele MOBs erstellt werden, wie freier Speicherplatz im Commodore zur Verfügung steht. Auf dem Bildschirm können jedoch nur 8 MOBs gleichzeitig dargestellt werden. Es ist möglich, in einem Programm einen MOB durch einen anderen zu ersetzen, indem man unter der gleichen Startadresse einen neuen MOB entwirft.

5.2.2 @ (Klammeraffe)

Format: @ .....
oder: @ .....
Ziel: Die Form eines MOBs eingeben.

Ein MOB besteht entweder aus einer Matrix von 24 x 21 (High Resolution) oder 12 x 21 (Low Resolution) Punkten. Der Befehl @ bietet eine sehr einfache Möglichkeit, einen MOB zu gestalten.
Jeder MOB besteht aus 21 BASIC-Zeilen mit dem Format @ ....., wobei jede Zeile aus einem @-Zeichen und 24 bzw. 12 Punkten besteht. Jeder Punkt stellt dabei einen Punkt der Matrix eines MOBs in der Farbe des Bildschirm-Hintergrundes dar. Ersetzt man diesen Punkt durch einen bestimmten Buchstaben, so ändert sich die Farbe dieses Punktes in der Matrix.

5.2.3 CMOB

Format: CMOB c1, c2
Ziel: Festlegen der Farben für Multi-Color-MOBs.

Der CMOB-Befehl legt die beiden zusätzlichen Farben für alle Multi-Color-MOBs fest. Die Parameter c1 und c2 können Werte zwischen 0 und 15 annehmen und entsprechen den Farben der Tabelle aus Abschnitt 3.5.2
Alle Punkte, die durch den Buchstaben B dargestellt werden, nehmen die Farbe entsprechend dem Parameter c1 an; die Punkte, die durch ein D dargestellt werden, entsprechen dem Parameter c2.

5.2.4 MOB SET

Format: MOB SET mb, blk, col, pr, res
Ziel: Festlegen, wie der MOB aufgebaut ist.

Der Befehl MOB SET legt bestimmte Eigenschaften fest, die einen MOB bestimmen.
mb = Nummer des MOBs, kann Werte von 0 bis 7 annehmen. Wenn mehrere MOBs auf dem Bildschirm sind, bewegt sich der MOB mit der niedrigeren Nummer vor dem MOB mit der höheren Nummer.
blk = legt fest, aus welchem Block die Daten für den MOB entnommen werden sollen (siehe Abschnitt 5.2.1).
col = legt die Farbe für den betreffenden MOB fest. Der Parameter kann entsprechend einer Farbe Werte zwischen 0 und 15 annehmen.

Bedeutung bei High-Resolution-MOBs:
Punkte der Matrix, die mit einem B (siehe Abschnitt 5.2.2) gekennzeichnet sind, werden mit der festgelegten Farbe ausgefüllt.
Bedeutung bei Multi-Color-MOBs:
Punkte der Matrix, die mit einem C (siehe Abschnitt 5.2.2) gekennzeichnet sind, werden mit der festgelegten Farbe ausgefüllt.
pr = legt die Priorität des MOB fest
pr=0 : Der MOB bewegt sich vor den Bildschirmzeichen.
pr=1 : Der MOB bewegt sich hinter den Bildschirmzeichen.
res =
res=0 : High-Resolution-MOB
res=1 : Multi-Color-MOB

5.2.5 MMOB

Format: MMOB mn, x1, y1, x2, y2, exp, speed

Ziel: Darstellen und/oder Bewegen eines MOBs.

Der Befehl MMOB bewirkt, daß ein MOB auf dem Bildschirm dargestellt wird und ermöglicht eine Bewegung dieses MOBs.
mn = gibt die Nummer des MOBs an, der dargestellt werden soll.
x1,y1 = Start-Koordinaten, von denen aus sich der MOB bewegt, bzw. dargestellt wird.
x2,y2 = Ziel-Koordinaten, auf die sich der MOB zubewegt.
exp = gibt die Größe des MOB an.
speed = Geschwindigkeit, mit welcher sich der MOB bewegt.
speed=1 : schnellste Geschwindigkeit
speed=255 : langsamste Geschwindigkeit

5.2.6 RLOCMOB

Format: RLOCMOB mn, x, y, exp, speed
Ziel: Bewegen eines MOBs.

RLOCMOB bewegt einen auf dem Bildschirm dargestellten MOB zu einer anderen Position auf dem Bildschirm. Die Koordinaten x,y bestimmen die Zielposition. Alle anderen Parameter entsprechen den Parametern von Abschnitt 5.2.5.

5.2.7 MOB OFF

Format: MOB OFF mn
Ziel: Löschen eines MOBs vom Bildschirm.

Der Parameter mn gibt die Nummer des MOBs an, der gelöscht werden soll.

5.2.8 DETECT

Format: DETECT n
Ziel: Abfrage der MOB-Kollision vorbereiten.

Der DETECT-Befehl muß vor dem Befehl CHECK im Programm kommen. Ist der Parameter n=0, so wird die Abfrage auf eine Kollision zwischen zwei MOBs vorbereitet. Ist n=1, so wird die Kollision zwischen Zeichen auf dem Bildschirm und einem MOB vorbereitet. Der Befehl DETECT muß zweimal gegeben werden. Bei der ersten Ausführung wird das Sprite-Kollisionsregister gelöscht; bei der zweiten Ausführung ist das Register bereit, eine Kollision zu registrieren.

5.2.9 CHECK

Format: IF CHECK (mn1, mn2) = 0 then Anweisung
oder: IF CHECK (0) = 0 THEN Anweisung
Ziel: Abfrage auf MOB-Kollision.

Der Befehl CHECK fragt ab, ob eine Kollision stattgefunden hat. Die Parameter mn1 und mn2 geben an, zwischen welchen beiden MOBs die Abfrage gemacht werden soll. Steht in den Klammern eine "0", dann wird abgefragt, ob ein MOB mit einem Bildschirmzeichen kollidiert ist. Hat eine Kollision stattgefunden, so wird die Anweisung hinter THEN ausgeführt.

5.3.Erstellen eines neuen Zeichensatzes

5.3.1 MEM

Format: MEM
Ziel: Verlegen des Zeichensatzes vom ROM in den RAM-Bereich.

Der Zeichensatz des Commodore ist in einem ROM festgelegt. Um diesen Zeichensatz neu zu definieren oder zu verändern, muß man ihn in den RAM-Bereich legen.Der Befehl MEM führt folgende Funktionen aus:
- Verschieben des Zeichensatz-ROM-Inhaltes in den RAM-Bereich hinter KERNAL
- Verschieben des Bildschirm-RAM nach $C000
- Eingrenzen des MOB-Bereiches auf Block 192 bis 255 (siehe Abschnitt 5.2.1).

5.3.2 DESIGN

Format: DESIGN 2, $E000 + ch x 8
Ziel: Festlegen, welches Zeichen von dem neu erstellten Zeichen ersetzt wird.

Der DESIGN-Befehl legt fest, welches Zeichen durch ein neu definiertes Zeichen ersetzt wird. Der Parameter ch muß den POKE-Code des zu ersetzenden Zeichens annehmen (siehe Handbuch). Das neu definierte Zeichen besteht aus einer Matrix von 8 x 8 Punkten. Wie die Matrix festgelegt wird, ist im folgenden Abschnitt erklärt.

5.3.3 @ (Klammeraffe)

Format: @ .....
Der Befehl @ ermöglicht die einfache Gestaltung eines neuen Zeichens. Nach dem @-Zeichen setzt man 8 Punkte. Da ein Zeichen aus einer Matrix von 8 x 8 Punkten besteht, sind 8 BASIC-Zeilen des Befehls @ ..... nötig, um ein neues Zeichen zu definieren. Jeder Punkt entspricht dabei einem Punkt der Matrix in der Bildschirm-Hintergrundfarbe. Soll ein Punkt der Matrix die Farbe des Cursors annehmen, so muß der Punkt durch ein B ersetzt werden.

5.4 Strukturierte Programmierung

5.5 Bedingungen prüfen und Programmschleifen

5.5.1 IF ... THEN ... ELSE

Format: IF Bedingung THEN Anweisung für wahr; ELSE : Anweisung für unwahr
Ziel: Auf eine Bedingung prüfen und je nach "wahr" und "unwahr" eine Anweisung ausführen.

Dieser Befehl ist im Prinzip mit dem Befehl IF ... THEN ... vergleichbar.
Der Unterschied besteht darin, daß man zusätzlich die Möglichkeit hat, eine Anweisung zu geben, die ausgeführt wird, wenn die Bedingung nicht erfüllt ist.
ACHTUNG:
ELSE muß durch Doppelpunkte (:) vom vorhergehenden und nachfolgenden Zeichen abgetrennt werden.

5.5.2 REPEAT ... UNTIL

Format: REPEAT : Schleife : UNTIL Bedingung erfüllt
Ziel: Wiederholen einer Anweisung bis eine Bedingung erfüllt ist.

REPEAT ... UNTIL führt eine ähnliche Funktion wie eine FOR ... NEXT - Schleife im Standard-BASIC aus. REPEAT startet die Ausführung der Schleife. UNTIL prüft auf eine Bedingung. Die Schleife wird verlassen, wenn die Bedingung erfüllt ist.

5.5.3 RCOMP

Format: RCOMP : Anweisung für wahr
:ELSE : Anweisung für unwahr
Ziel: Abfrage nach der gleichen Bedingung des zuletzt ausgeführten IF...THEN...ELSE-Befehles.

RCOMP ermöglicht es, die Bedingung in der zuletzt ausgeführten IF...THEN...ELSE-Abfrage zu prüfen.
Dies ist besonders dann eine Erleichterung, wenn die Bedingung aus mehreren Verknüpfungen besteht.

5.5.4 LOOP ... EXIT IF ... END LOOP

Format: LOOP : Programmschleife : EXIT IF Bedingung : END LOOP
Ziel: Durchlaufen einer Schleife, bis eine Bedingung erfüllt ist.

Dieser Befehl ermöglicht es, eine Programmschleife aufzubauen, in der die Bedingung des Schleifenendes festgelegt ist. Es können mehrere Bedingungen in der Schleife vorkommen. Ist eine Bedingung erfüllt, dann wird die Schleife verlassen und der Befehl hinter der Programmschleife ausgeführt. Ist keine Bedingung erfüllt, so wird die Schleife erneut durchlaufen.

5.6 Programmiertechnik

5.6.1 PROC

Format: PROC Label
Ziel: Vergeben einer symbolischen Sprungadresse (Label).

PROC bietet die Möglichkeit, einer Programmroutine eine symbolische Sprungadresse (Label) zuzuweisen. Alle Zeichen hinter dem Befehl PROC bilden den Label, unter dem die Routine dann aufgerufen werden kann. In dieser Programmzeile dürfen keine weiteren Anweisungen vorkommen.

5.6.2 END PROC

Format: END PROC
Ziel: Das Ende einer Routine festlegen.

END PROC legt das Ende einer geschlossenen Routine fest. Der Befehl ist mit dem RETURN des Standard-BASIC vergleichbar. Er bewirkt den Rücksprung auf den unmittelbar hinter EXEC folgenden Befehl.

ACHTUNG:
END PROC darf nur hinter EXEC ausgeführt werden, da sonst die Fehlermeldung "?END PROC WITHOUT EXEC IN..." kommt.

5.6.3 CALL

Format: CALL Label
Ziel: Sprung zu der symbolischen Sprungadresse (Label).

Der Befehl CALL entspricht dem Befehl GOTO des Standard-BASIC. Hier wird aber nicht in eine feste Zeilennummer gesprungen, sondern zu einer symbolischen Sprungadresse (Label).

5.6.4 EXEC

Format: EXEC Label
Ziel: Unterprogrammaufruf.

Der EXEC-Befehl entspricht dem GOSUB-Befehl des Standard-BASIC. Das Unterprogramm wird hier aber unter einem symbolischen Namen (Label) aufgerufen. Nach dem Rücksprung aus dem Unterprogramm wird der Befehl unmittelbar hinter EXEC ausgeführt.

5.7 Variablen

5.7.1 LOCAL

Format: LOCAL Variable 1, Variable 2, Variable 3, ...
Ziel: Festlegen der Variablen an einer bestimmten Stelle im Programm.

Der LOCAL-Befehl weist den Variablen neue Werte zu. Der ursprüngliche Wert dieser Variablen bleibt erhalten.

5.7.2 GLOBAL

Format: GLOBAL
Ziel: Wiederherstellen der ursprünglichen Wertzuweisung im Programm.

Der GLOBAL-Befehl stellt den ursprünglichen Zustand aller Variablen wieder her, die mit dem Befehl LOCAL verändert wurden.

5.8 Fehler-Behandlung

5.8.1 ON ERROR

Format: ON ERROR : GOTO Zeilennummer
Ziel: Abfangen eines Programmfehlers.

Der Absturz eines Programmes kann verhindert werden. Tritt ein Fehler auf, so springt das Programm mit Hilfe dieses Befehls in die angegebene Zeilennummer. In der Variablen ERRN wird die Fehlernummer und in ERRLN die Zeilennummer, in der sich der Fehler befindet, gespeichert.
Folgende Fehler können mit Simon's BASIC abgefangen werden:

1. TOO MANY FILES,
2. FILE OPEN,
3. FILE NOT OPEN,
4. FILE NOT FOUND,
5. DEVICE NOT PRESENT,
10. NEXT WITHOUT FOR,
11. SYNTAX,
12. RETURN WITHOUT GOSUB,
13. OUT OF DATA,
14. ILLEGAL QUANTITY,
15. OVERFLOW,
16. OUT OF MEMORY,
17. UNDEFINED STATEMENT,
18. BAD SUBSCRIPT,
19. RE-DIMENSIONED ARRAY,
20. DIVISION BY ZERO,
21. ILLEGAL DIRECT,
22. TYPE MISMATCH,
23. STRING TOO LONG.

5.8.2 NO ERROR

Format: NO ERROR
Ziel: Fehlermeldung unterdrücken.

Mit dem Befehl NO ERROR wird eine Fehlermeldung unterdrückt. Die Fehlerbehandlung des ON ERROR-Befehls wird gleichzeitig abgebrochen. Das Programm fährt fort. Tritt ein weiterer Fehler auf, so kommt eine Fehlermeldung des Standard-BASIC.

5.8.3 OUT

Format: OUT
Ziel: Abschalten der SIMONS´ BASIC Fehlerbehandlung.

Nach dem Befehl OUT unterliegt die Kontrolle der Fehlerbehandlung wieder dem Commodore, d.h., es erscheinen die üblichen Fehlermeldungen des Standard-BASIC.

5.9 Musik machen mit SIMONS´ BASIC

6.0 Musik-Befehle

6.0.1 VOL

Format: VOL n
Ziel: Lautstärke einstellen.

Der VOL-Befehl stellt die Lautstärke der drei Tongeneratoren ein. Der Parameter n kann Werte von 0 bis 15 annehmen.
n=0 schaltet die Tongeneratoren ab, n=15 ist die maximale Lautstärke.

6.0.2 WAVE

Format: WAVE, ST, Binärziffer

Ziel: Einstellen der Wellenform.

Die Wellenform bestimmt den Klangcharakter der drei Tongeneratoren (Stimmen) des Commodore.
ST=1 : Stimme 1,
ST=2 : Stimme 2,
ST=3 : Stimme 3
Der zweite Parameter im WAVE-Befehl ist eine Binärziffer. Durch Setzen der einzelnen Bits (1) wird der Charakter des Tones eingestellt.
Bit 0 setzt das Gate-Bit (nicht erforderlich)
Bit 1 setzt die Synchronisation
Bit 2 setzt die Ringmodulation
Bit 3 setzt ein Test-Bit (sollte nicht gesetzt werden)
Bit 4 setzt die Dreieck-Wellenform
Bit 5 setzt die Sägezahn-Wellenform
Bit 6 setzt die Rechteck-Wellenform
Bit 7 setzt das Rauschen

6.0.3 ENVELOPE

Format: ENVELOPE
Ziel: Einstellen der Hüllkurve.

Die Hüllkurve jeder Stimme kann getrennt eingestellt werden. Sie besteht aus vier Parametern: Attack, Decay, Sustain, Release.

6.0.4 MUSIC

Format: MUSIC n, "Noten"
oder: MUSIC n, Stringvariable
Ziel: Festlegen der Noten, welche gespielt werden sollen.

In dem Befehl MUSIC werden weitere Parameter festgelegt, die den entstehenden Ton beeinflussen. Der Parameter n bestimmt die Tondauer und muß im Bereich von 1 bis 255 liegen. Der nächste Parameter legt die Tonhöhe und -dauer einer oder mehrerer Noten fest. Zunächst wird in dem String nach SHIFT CLR/HOME die Stimme (1-3) festgelegt. Danach folgt eine Steuertaste (F1-F8) für den Takt und die Note (C0-A7).

Funktionen der Steuertasten:
F1=1/16tel-Note,
F3=1/8,
F5=1/4,
F7=1/2,
F2=1/1,
F4=2/1,
F6=4/1,
F8=8/1.

Jede Note besteht aus einem Buchstaben (C-H) und einer Oktave (0-7). Man erhält die um einen halben Ton erhöhte Note, wenn man gleichzeitig mit dem Notennamen die SHIFT-Taste drückt. Am Ende jedes Noten-Strings muß SHIFT CLR/HOME und G eingegeben werden. Dies startet den RELEASE-Abschnitt der letzten Note.

6.0.5 PLAY

Format: PLAY n
Ziel: Wiedergabe der Musik.

Der PLAY-Befehl startet das Spielen der komponierten Musik. Der Parameter n bezieht sich auf die Ausführung des weiteren Programms.
n=0: Musik aus
n=1: spielt die Musik und wartet mit der weiteren Ausführung des Programms.
n=2: spielt die Musik und fährt mit der Programmausführung fort.

6.1 Funktionen für Lightpen, Joystick und Paddle

6.2 PENX

Format: Variable = PENX

Ziel: Bestimmen der X-Koordinate des Lightpen.

PENX gibt die Position des Lightpen relativ zum linken Bildschirmrand an (0 bis 320).

6.3 PENY

Format: Variable = PENY
Ziel: Bestimmen der Y-Koordinate des Lightpen.

PENY gibt den Abstand vom oberen Bildschirmrand an (0 bis 200).

6.4 POT

Format: Variable = POT (0)
oder: Variable = POT (1)
Ziel: Den Widerstand des Paddle feststellen.

Die POT-Funktion ermöglicht es, die Stellung der Paddles festzustellen. Das Ergebnis liegt im Bereich von 0 bis 255. Die Ziffer hinter dem POT-Befehl gibt an, welches Paddle abgefragt werden soll.

6.5 JOY

Format: Variable = JOY
Ziel: Die Funktionen des Joysticks bestimmen.

Mit dem JOY-Befehl kann man feststellen, in welche Richtung der Joystick zeigt, oder ob der Feuerknopf gedrückt wurde.

*15-01-90/01/E

Technische Änderungen vorbehalten!

Alle Rechte, auch Übersetzungen, vorbehalten. Reproduktionen gleich welcher Art, ob Fotokopie, Mikrofilme oder Erfassung in Datenverarbeitungsanlagen, nur mit schriftlicher Genehmigung durch CONRAD ELECTRONIC GmbH.

© Copyright 1990 by CONRAD ELECTRONIC GmbH, 8452 Hirschau.

*15-01-90/01/E